Cocoa Text

Session 306

Cocoa Text

Doug Davidson
Cocoa Frameworks Engineer

Introduction and Concepts

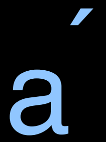
- Text system is at the heart of Cocoa
- Responsible for almost all visible text
- Handles everything from simple string display to the five-minute text editor, and beyond
- Fully Unicode based
- Highly customizable

Concepts

- Characters + attributes = attributed string
- Attributes include fonts, paragraph attributes
- Glyphs are elements of fonts
- Containers describe geometry

Example:

- Character: Latin small letter a acute (á)
- Attributes: Helvetica 64, blue
- Glyphs: a ´
- Displayed result:

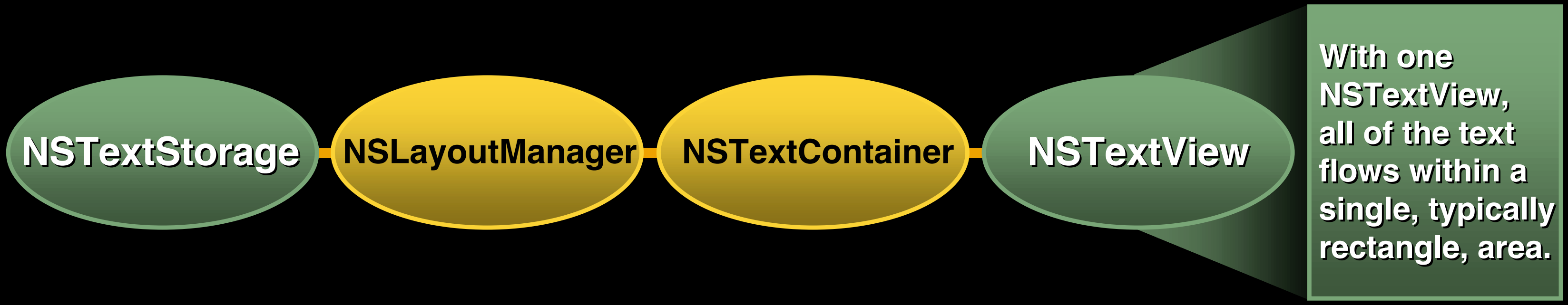


Processes

- Attribute fixing—make sure attributes are OK
- Glyph generation—from characters to glyphs
- Layout—position glyphs in containers
- Display—send glyphs to Quartz for display

All of these usually done lazily on demand

Text System Classes



With one NSTextView, all of the text flows within a single, typically rectangle, area.

Text System Classes

- Model—View—Controller
- Model classes model characters, attributes, geometry
- Controller classes control glyph generation and layout
- View classes handle user input and display

Model Classes

- NSTextStorage is NSMutableAttributedString
- NSFont, NSColor, NSParagraphStyle, NSTextTab
- NSTextAttachment models attached file
- NSTextContainer models geometry of layout

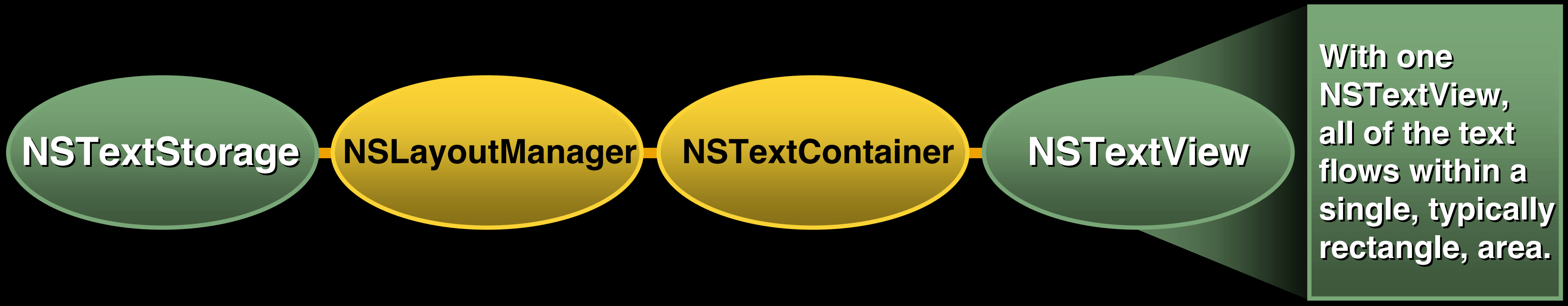
Controller Classes

- NSLayoutManager is the boss
- Workhorse class that manages all the rest
- Controls glyph generation and layout
- Performs actual display of glyphs
- Source for information about glyphs and layout
- Calls on NSTypesetter to perform layout

View Classes

- NSTextView handles display and input for a single NSTextContainer region
- NSText is vestigial abstract superclass
- Multiple NSTextViews work together
- Ruler classes handle text rulers
- NSTextAttachmentCell draws attachment

Class Relationships

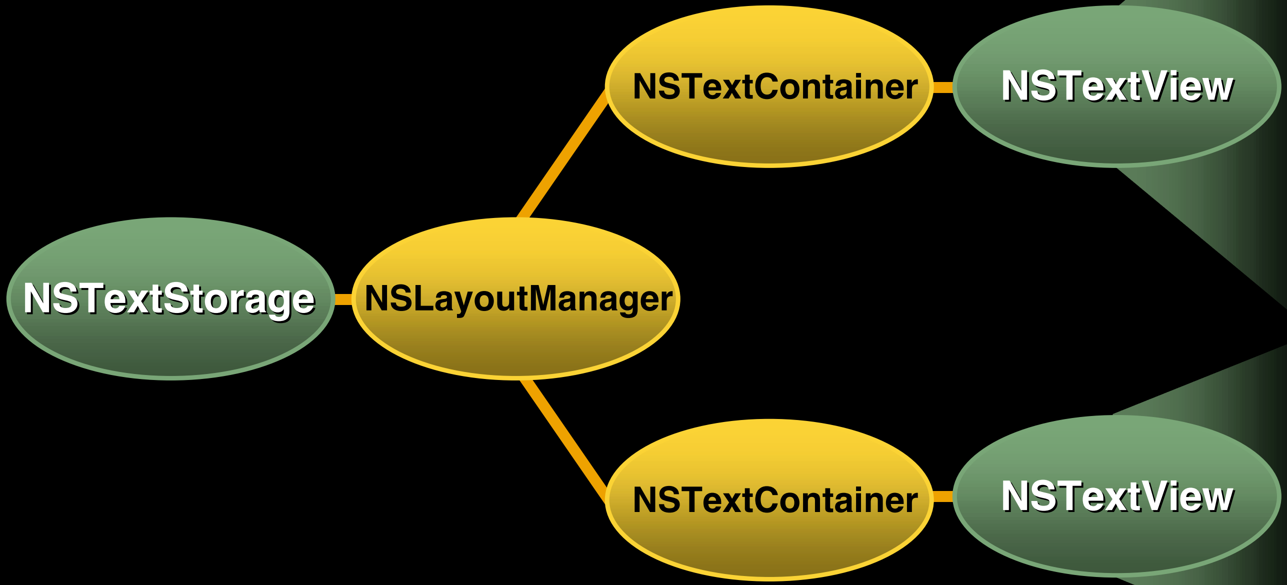


With one NSTextView, all of the text flows within a single, typically rectangle, area.

Class Relationships

- One text storage may have multiple layout managers, corresponding to different layouts
- One layout manager may have a list of text containers, into which the text flows
- A text container has at most one text view
- Usually storage owns layout owns container owns view, but there is a simplified option

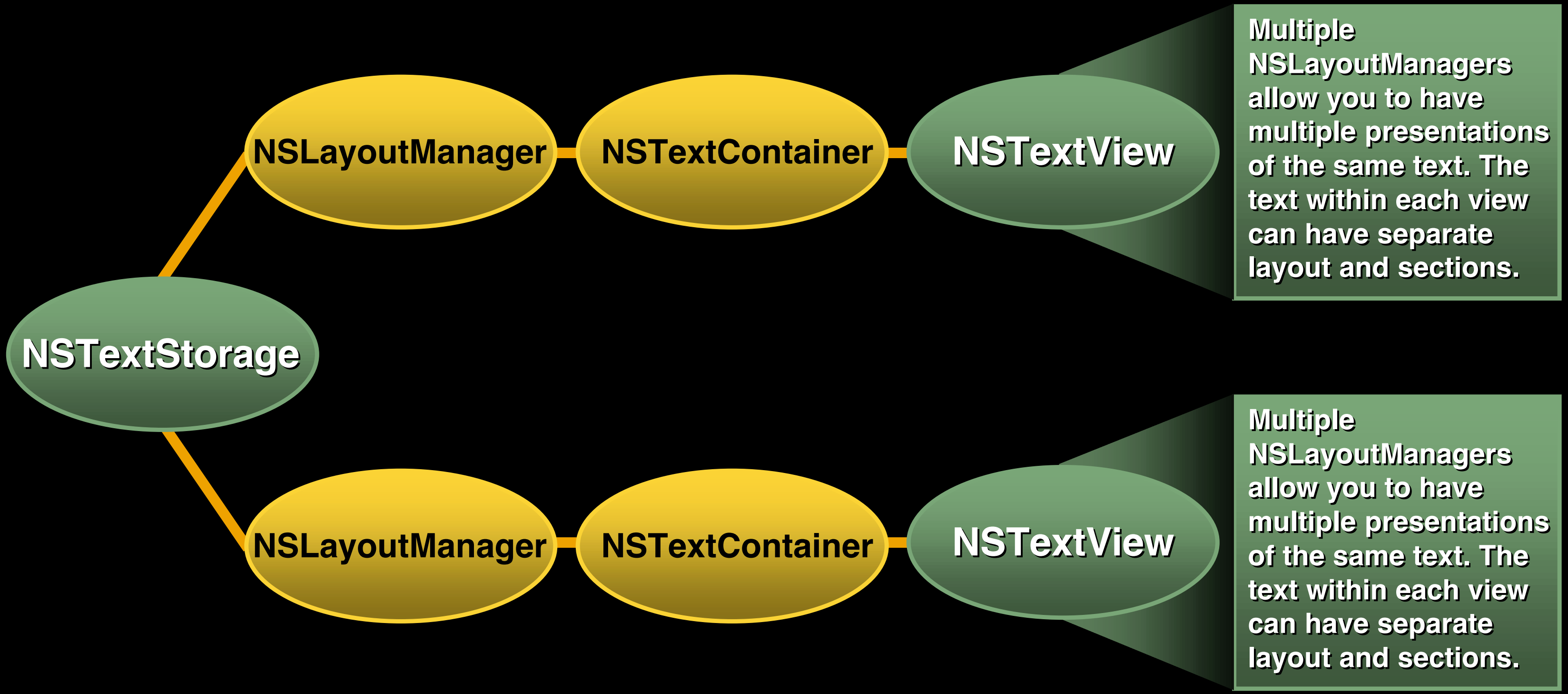
Class Relationships



As text is added, it fills the region defined by the first NSTextContainer. The text is displayed in the NSTextView that is paired with the NSTextContainer. When there's no more room, another

NSTextContainer (and associated NSTextView) is Added, and the Text flows onto the second page.

Class Relationships



Multiple NSLayoutManagers allow you to have multiple presentations of the same text. The text within each view can have separate layout and sections.

Multiple NSLayoutManagers allow you to have multiple presentations of the same text. The text within each view can have separate layout and sections.

Simple Text System Usages

- String drawing for string or attributed string
- Cell drawing used by controls
- For editing, control uses a shared layout manager and text view, the “field editor”
- Can use layout manager without text view to measure and/or draw text

Simple Text View Usages

- Simplest and most common case is one text view, one text container, one layout
- Rule of thumb: Use notification/delegation before subclassing
- Text view’s delegate has extensive control over handling of user input

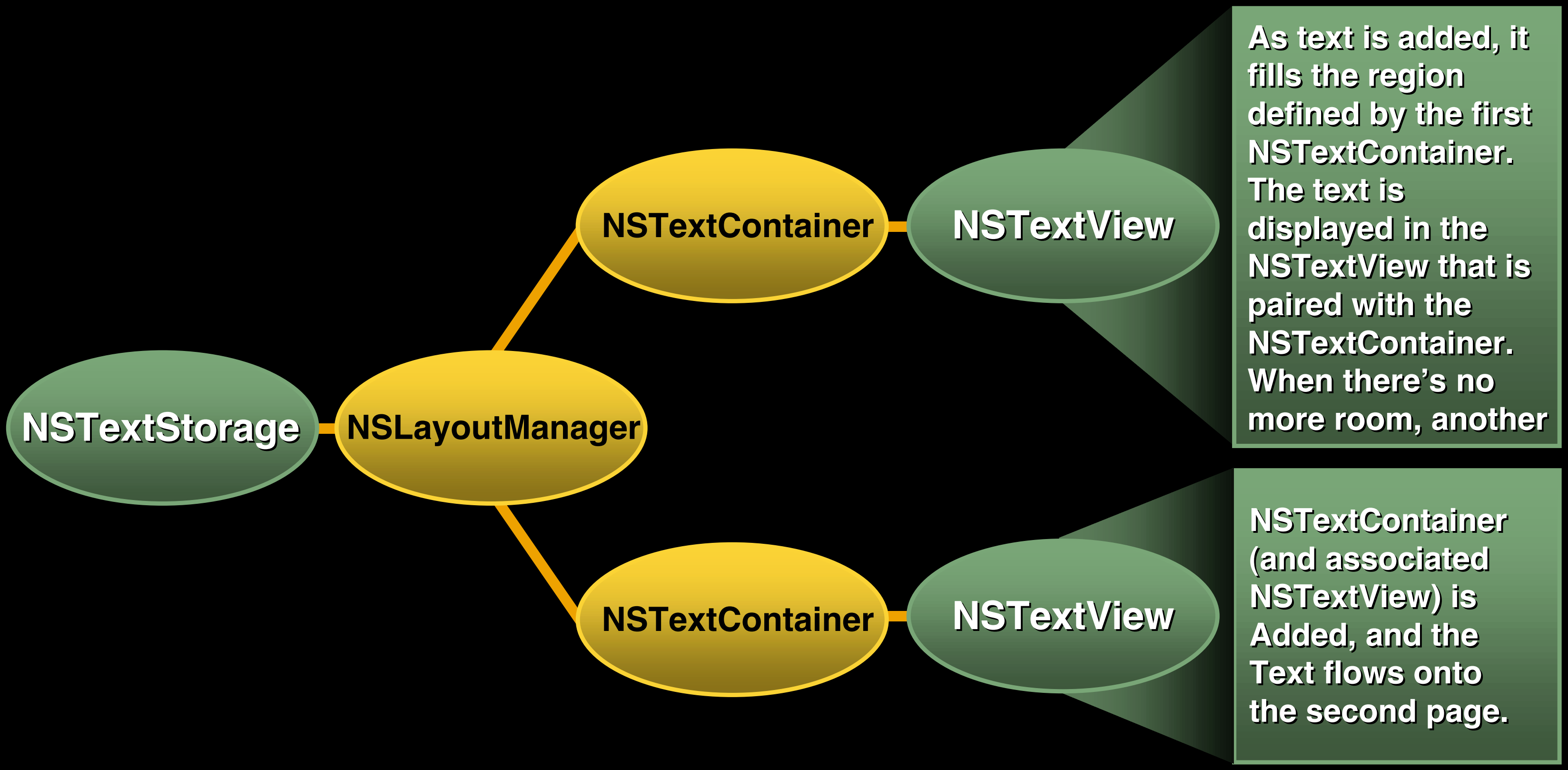
Demo

Text View Delegate

Beyond the Single Text View

- Multiple text containers: Multiple pages, multiple columns, arbitrary regions
- Text container subclass can describe more or less arbitrary geometry
- Layout manager has ordered list of containers into which it lays text
- One text view per container, with shared state

Multiple Text Containers



As text is added, it fills the region defined by the first NSTextContainer. The text is displayed in the NSTextView that is paired with the NSTextContainer. When there's no more room, another

NSTextContainer (and associated NSTextView) is Added, and the Text flows onto the second page.

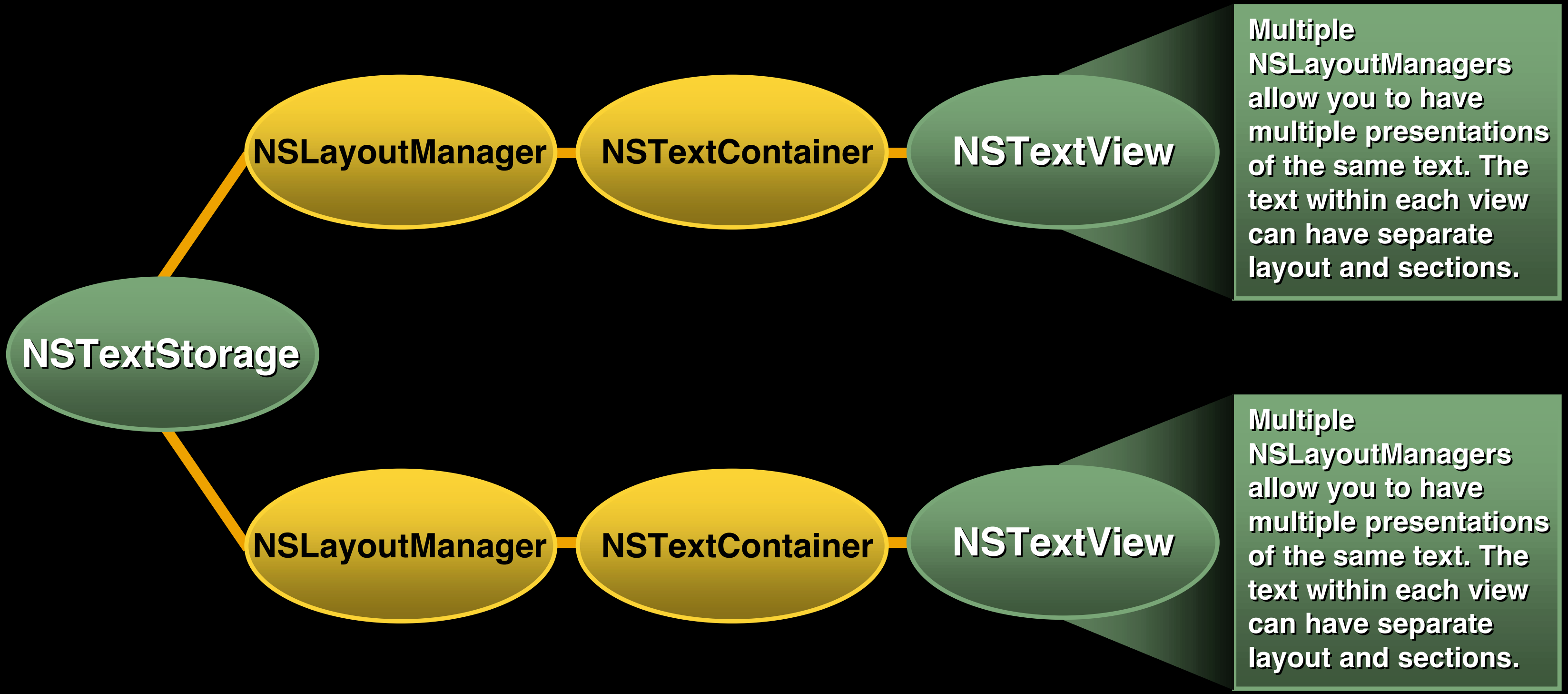
Multiple Layouts

- A single text storage may have multiple layout managers, for multiple views of the same text
- For example, a slide presentation app might display text in a single container (outline), and simultaneously in multiple containers (slides)
- Changes made in one take effect in text storage, which notifies all layout managers, which invalidate appropriate ranges

Multiple Layouts (Cont.)

- Layout manager also sets needs display for appropriate regions of views
- When views are redisplayed, they ask for glyph information and glyph drawing
- This lazily causes attribute fixing, glyph generation, and layout as needed

Multiple Layouts



Multiple NSLayoutManagers allow you to have multiple presentations of the same text. The text within each view can have separate layout and sections.

Multiple NSLayoutManagers allow you to have multiple presentations of the same text. The text within each view can have separate layout and sections.

Text Options

- Container may resize to match view, or view to match container, in horizontal or vertical
- Text views often in scroll view, which may scroll horizontally or vertically
- Container can be inset within text view, and text can be inset within text container

Text Options (Cont.)

- Text view has options to control selectability, editability, rich text, attachments, rulers, font panel, spellchecking, smart cut/paste, selected attributes, field editor behavior
- Layout manager has options to control background layout, hyphenation, attachment scaling, use of screen fonts

Assembling the Pieces

- Create text storage with text
- Create layout manager(s) and add to text storage
- Create text containers and add to layout manager
- Create text views, if desired, and set for text containers
- Size and pad containers, inset them in views, and position views in superview

Demo

Text Options

Dealing With the Text

- NSTextStorage stores the text, as a mutable attributed string subclass
- Use mutable attributed string methods to alter it
- Special feature of text storage is that it notifies layout managers of changes
- For efficiency, wrap sequences of changes in beginEditing/endEditing to coalesce

Text Attributes

- Recognized attributes are listed in AppKit's NSAttributedString.h header
- Font, colors, underlines, sub/superscript, ligatures, kerning, baseline offset, paragraph style
- Bold, italic, etc. are traits of the font, managed by NSFontManager
- Margins, tabs, spacing are part of paragraph style
- New: Right, center, decimal tabs

Special Attributes

- Attachments consist of attachment character with attachment attribute
    - Attribute value is NSTextAttachment, which points to an NSFileWrapper
    - Attachment cell will automatically be created, or you can use your own
- Link attribute can point to URL or other object of your choice

Attribute Fixing

- Attribute fixing performed lazily on demand by text storage
    - Font fixing substitutes fonts to make sure characters can be displayed
    - Paragraph fixing makes sure whole paragraph has the same paragraph style
    - Attachment fixing makes sure attachment attributes go with attachment characters

Getting Text In and Out

- Native format is RTF, or RTFD with attachments
- For pasteboard we provide these, plus plain text
- Methods for reading and writing RTF and RTFD
- Methods for reading from a file, which may be plain text, RTF, RTFD, HTML, SimpleText, etc.
- Filter services allow third-party extensions

Services

- Standard services allow processing of text
- Text provided and returned on pasteboard
- Filter services for images and text
- Spellcheckers are also a form of service
    - All-new spellcheckers for Jaguar

Archiving

- New keyed archiving supports archiving of most classes in the text system
- NSTextView, NSLayoutManager, NSTextContainer, and NSTextStorage archive
- Archiving preserves all attributes in text storage
    - Custom attributes should be archivable

Dealing With Layout

- NSLayoutManager answers your questions about glyphs and layout
- Glyph generation and layout performed lazily on demand
- Layout manager observes changes to text storage and invalidates as necessary
    - You can also invalidate manually

Characters and Glyphs

- Layout manager maps glyph ranges to character ranges
- Layout manager stores sequence of glyphs
  - Control glyph used for tabs, line breaks, etc.
  - Null glyph sometimes used for padding
- Layout manager stores glyph attributes
  - elastic, inscription

Layout Information

- Layout manager contacts typesetter, which performs layout
- Typesetter may insert or change glyphs, or make glyphs not shown
- Typesetter contacts text container for geometry
- Typesetter informs layout manager which container a glyph goes in, which line fragment, and where in the fragment

Ask the Layout Manager

- Position, size, and glyph range for line fragment (position in text container coordinates)
- Glyph range for text container
- Text container, line fragment, and position in fragment for glyph
- Glyph for coordinates, bounding rect for glyph range, rect array for glyph range

Drawing Glyphs

- Layout manager actually sends the glyphs to be drawn, and draws backgrounds and underlines (using Quartz)
- You can request drawing yourself
- Must have locked focus in appropriate view
- Positions are relative to text container

Demo

Layout Manager

Subclassing Text Objects

- First, check to see if you can use existing options
  - Example: Selected text attributes
  - Example: Temporary attributes
- Next, try notification, delegation, categories
  - Example: Command by selector
- Override minimal methods to do what you want

Subclassing NSTextStorage

- Subclass to provide your own storage mechanism
- Subclass to do your own attribute fixing
- Subclass to obtain additional notifications

Subclassing NSTextContainer

- Subclass to provide custom geometry
- Your custom container will be passed in a proposed line rectangle
- Return an adjusted rectangle, plus an additional rectangle for possible further fragments within the same line

Subclassing NSTextView

- Subclass to alter user interaction
- Subclass to alter drawing behavior at the view level
- Subclass to alter context menus, cut and paste, drag and drop, etc.

Subclassing NSLayoutManager

- Subclass to provide custom drawing behavior
  - Background, glyphs, underlines
- Subclass to store additional glyph attributes
- Subclass to override hit testing, selection rectangles, etc.

NSTextAttachmentCell

- Standard text attachment models attached file, with standard cell to provide inline image
- Subclass text attachment cell to do arbitrary custom drawing
- Subclass to handle mouse clicks within the cell

Subclassing NSTypesetter

- Most difficult class to subclass
- Hooks after layout of glyph, line
- Subclass to alter layout at the level of individual glyphs in a line fragment

Demo

Subclassing

Where to Go From Here

- /Developer/Examples/AppKit
  - TextEdit
  - TextSizingExample
  - CircleView
- http://developer.apple.com

  Documentation > Mac OS X

  > Cocoa > Text Handling

Cocoa Documentation

- Object-Oriented Programming and the Objective-C Language
- Programming Topics

Application Architecture	Memory Management
Foundation Framework	Multithreading
Loading Resources	Notifications

. . .and many more!

Documentation > Cocoa
developer.apple.com/techpubs/macosx/Cocoa/CocoaTopics.html

For More Information

- O'Reilly "Learning Cocoa" and "Building Cocoa Applications: A Step-by-Step Guide"
- Cocoa Developer Documentation
  http://developer.apple.com/techpubs/macosx/Cocoa/CocoaTopics.html
- Apple Customer Training
  http://train.apple.com/

Roadmap

300 Introduction to Cocoa: What's Cocoa?	Room A1 **Mon., 5:00pm**
301 Cocoa: What's New: New features and API since 10.1	Civic **Tues., 9:00am**
302 Cocoa API Techniques: Understanding, leveraging, and extending	Hall 2 **Thurs., 9:00am**
305 Cocoa Drawing: Drawing using Cocoa APIs	Hall 2 **Fri., 10:30am**
FF016 Cocoa: Comments and suggestions for Cocoa	Room A1 **Fri., 5:00pm**

Who to Contact

Heather Hickman
Cocoa Technology Manager
hhickman@@apple.com

Cocoa Feedback
cocoa-feedback@group.apple.com

Cocoa Development Mailing List
Subscribe at
www.lists.apple.com/mailman/listinfo/cocoa-dev

Q&A

Heather Hickman
Cocoa Evangelist
hhickman@apple.com

http://developer.apple.com/wwdc2002/urls.html

WWDC2002

WWDC2002

WWDC2002